SPECIFICATIONS TECHNIQUES

I. Extraction des données brutes – Base de données

Les données utilisées sont issues de l'*infobox* de la page du site Wikipédia dédiée à chaque pays. Pour éviter une requête systématique du client au(x) serveur(s) Wikipédia qui, en cas de fort afflux, produirait l'effet d'une DDOS, le choix a été fait de télécharger une fois pour toutes les *infobox* dans l'application côté serveur, sous forme de fichiers *.json*. La structure naturelle pour le stockage et la lecture de ces fichiers est une base de données, établie au format SQL : les requêtes pour la produire et y accéder sont effectuées sous Python3, grâce au module *Sqlite3*. Bien entendu, la mise à jour des données est possible : il suffit de télécharger les *infobox* Wikipédia les plus récentes, et de réexécuter le script de création de la BDD.

La création de la base de données requiert le lancement du script Python « creation_BDD.py ». Pour éviter que la lecture et le débogage ne soient indigestes, les tâches à réaliser sont décomposées en fonctions et sous-fonctions, chacune prenant en charge une tâche bien spécifique précisée par son nom :

- Récupération d'une donnée d'un pays dans les fichiers *.json* : préfixe « **get_** »
- Sauvegarde d'une donnée pour tous les pays dans la base de données : préfixe « **save_** »
- Destruction d'une donnée pour tous les pays dans la base de données : préfixe « **delete_** »

Suivant ces considérations, le programme se déroule en trois grandes phases :

1) Définition des fonctions

- De récupération des données :

  Paramètres d'entrée (excepté cv_coords) :

  → *wp_info* : dictionnaire (*dict*) contenant les informations du pays. Il est directement issu de la lecture du fichier *.json*. Les fonctions sont décrites ci-dessous par table de la BDD.
  → *pays* : chaîne de caractères (*str*). Paramètre parfois non nécessaire.

  ➔ 'GLOBAL' :

  - « get_names » : nom commun et nom conventionnel d'un pays (sous réserve d'existence).
    Format de renvoi : liste de deux chaînes de caractères *str*.
    NB : si l'un des deux noms n'existe pas, on affecte la valeur de l'autre nom, ou bien de l'intitulé du fichier *.json* du pays.

- « get_capitale » : capitale du pays.
  Format de renvoi : chaîne de caractères *str*.

- « get_flag » : intitulé de l'image PNG du drapeau renfermée dans le dossier compressé *flag.zip* contenu dans *Serveur*.
  Format de renvoi : chaîne de caractères *str*.
  NB : l'affichage du drapeau est effectué par le client via une requête HTML.

- « get_coords » : coordonnées de la capitale d'un pays (sous réserve d'existence).
  Format de renvoi : dictionnaire avec deux clés en chaînes de caractères *str*, deux valeurs flottantes *flott*. Exemple : *{'lat'=10.4, 'lon'=250.6669}*.
  NB : la fonction appelle « *cv_coords* » en interne pour convertir au format numérique.

- « cv_coords » : conversion d'une chaîne de caractères décrivant la position d'une capitale en dictionnaire Python de latitude et longitude.
  Format d'entrée : chaîne de caractère *str* à mettre en forme.
  Format de renvoi : dictionnaire avec deux clés en chaînes de caractères *str*, deux valeurs flottantes *flott*. Exemple : *{'lat'=10.4, 'lon'=250.6669}*.

- « get_adresse_wiki » : adresse de la page Wikipédia de la capitale.
  Format de sortie : chaîne de caractère *str* de l'adresse de la page.
  NB : fonction inusitée pour le moment. Renseignée dans une version future de l'application. Pour le moment, l'adresse de la capitale est récupérée manuellement en 3).

➔ 'ECONOMIE' :

- « get_devise » : devise utilisée dans le pays.
  Format de renvoi : chaîne de caractères *str*.

- « get_PIB_nominal » : PIB nominal courant d'un pays en dollar américain.
  Format de renvoi : flottant *float*.

- « get_PIB_par_tete » : PIB nominal par tête d'un pays en dollar américain.
  Format de renvoi : flottant *float*.

➔ 'MISCELLANEOUS' :

- « get_sens_circulation » : sens de circulation pour les voitures (deux valeurs possibles, *right* ou *left*).
  Format de renvoi : chaîne de caractères *str*.

- « get_code_appel » : code d'appel à l'international du pays.

Format de renvoi : chaîne de caractères *str*.

- « get_domaine_internet » : domaine des sites internet du pays.
  Format de renvoi : chaîne de caractères *str*.

➔ 'POLITIQUE' :

- « get_nom_chef_etat » : nom du chef d'état, indépendamment de son titre.
  Format de renvoi : chaîne de caractères *str*.

- « get_type_chef_etat » : titre du chef d'état, indépendamment de son nom.
  Format de renvoi : chaîne de caractères *str*.

- « get_regime » : régime politique du pays, en lien avec le titre du chef d'état.
  Format de renvoi : chaîne de caractères *str*.

➔ 'DEMOGRAPHIE' :

- « get_habitants » : nombre d'habitants recensés ou estimés.
  Format de renvoi : chaîne de caractères *str*.
  NB : retourne 'inconnu' s'il n'est pas renseigné.

- « get_habitants_annee » : année pour laquelle le nombre d'habitants recensés ou estimés est donné. Complémentaire de « get_habitants ».
  Format de renvoi : chaîne de caractères *str*.
  NB : retourne 'inconnu' s'il n'est pas renseigné.

- De sauvegarde de données :

  Aucun renvoi (fonctions de type *void*).

➔ 'GLOBAL' :

Paramètres en entrée : conn, pays, continent, wp.
Respectivement : connecteur à la base de données (objet *Sqlite3.Connection*) ; nom du pays (chaîne de caractère) ; nom du continent (chaîne de caractère) ; informations du pays (dictionnaire).

« save_global » : sauvegarde des informations globales pour le pays renseigné dans la table 'GLOBAL'.

➔ ‘ECONOMIE’ :
Paramètres en entrée : conn, pays, wp.
Respectivement : connecteur à la base de données (objet *Sqlite3.Connection*) ; nom du pays (chaîne de caractère) ; informations du pays (dictionnaire).

« save_economie » : sauvegarde des informations économiques pour le pays renseigné dans la table ‘ECONOMIE’.

➔ ‘MISCELLANEOUS’ :

Paramètres en entrée : conn, pays, wp.
Respectivement : connecteur à la base de données (objet *Sqlite3.Connection*) ; nom du pays (chaîne de caractère) ; informations du pays (dictionnaire).

« save_miscellaneous » : sauvegarde des informations additionnelles (vrac) pour le pays renseigné dans la table ‘MISCELLANEOUS’.

➔ ‘POLITIQUE’ :

Paramètres en entrée : conn, pays, wp.
Respectivement : connecteur à la base de données (objet *Sqlite3.Connection*) ; nom du pays (chaîne de caractère) ; informations du pays (dictionnaire).

« save_politique » : sauvegarde des informations politiques du pays renseigné dans la table ‘POLITIQUE’.

➔ ‘DEMOGRAPHIE’ :

Paramètres en entrée : conn, pays, wp.
Respectivement : connecteur à la base de données (objet *Sqlite3.Connection*) ; nom du pays (chaîne de caractère) ; informations du pays (dictionnaire).

« save_demographie » : sauvegarde des informations démographiques du pays renseigné dans la table ‘DEMOGRAPHIE’.

➔ AFFICHAGE DE TOUTES LES DONNES :

- « print_all_info » : affiche les informations décommentées dans la fonction

➔ ENREGISTREMENT DE TOUTES LES DONNES :

Paramètre d’entrée : conn connecteur à la BDD (objet *Sqlite3.Connection*)

- « save_all_info » : sauvegarde des informations de tous les pays dans les tables respectives.
  NB1 : cette fonction itère sur les deux continents, puis sur chaque pays de chaque continent, les fonctions précédentes en préfixe « save ».

- De suppression des données :

  Ces fonctions établissent une requête SQL permettant de vider une table spécifique des données qu'elle contenait (si existence desdites données). Aucun renvoi.

  Paramètre d'entrée : conn connecteur à la BDD (objet *Sqlite3.Connection*)

  ➔ 'GLOBAL' :

  « delete_global » : supprime les informations pour tous les pays renseignés dans la table 'GLOBAL'.

  ➔ 'ECONOMIE' :

  « delete_economie » : supprime les informations pour tous les pays renseignés dans la table 'ECONOMIE'.

  ➔ 'MISCELLANEOUS' :

  « delete_miscellaneous » : supprime les informations pour tous les pays renseignés dans la table 'MISCELLANEOUS'.

  ➔ 'POLITIQUE' :

  « delete_politique » : supprime les informations pour tous les pays renseignés dans la table 'POLITIQUE'.

  ➔ 'DEMOGRAPHIE' :

  « delete_demographie » : supprime les informations pour tous les pays renseignés dans la table 'DEMOGRAPHIE'.

  ➔ ANNIHILATION DE TOUTES LES DONNES :

  Paramètre d'entrée : conn connecteur à la BDD (objet *Sqlite3.Connection*)

  - « delete_all_info » : destruction des informations de tous les pays dans les tables respectives.

2) Génération de la base de données :

- Exécution d'un « nettoyage » préalable : si une table existe déjà, elle est écrasée.
- Exécution de la fonction de création de la base de données complète.

3) Modifications manuelles :

Comme mentionné en 1), certaines informations sont inexistantes pour des pays particuliers. Il faut alors résoudre ces difficultés au cas par cas. La fonction « modifications_manuelles » est exécutée pour gérer tous les cas litigieux : valeurs manquantes, inexactes,… etc.

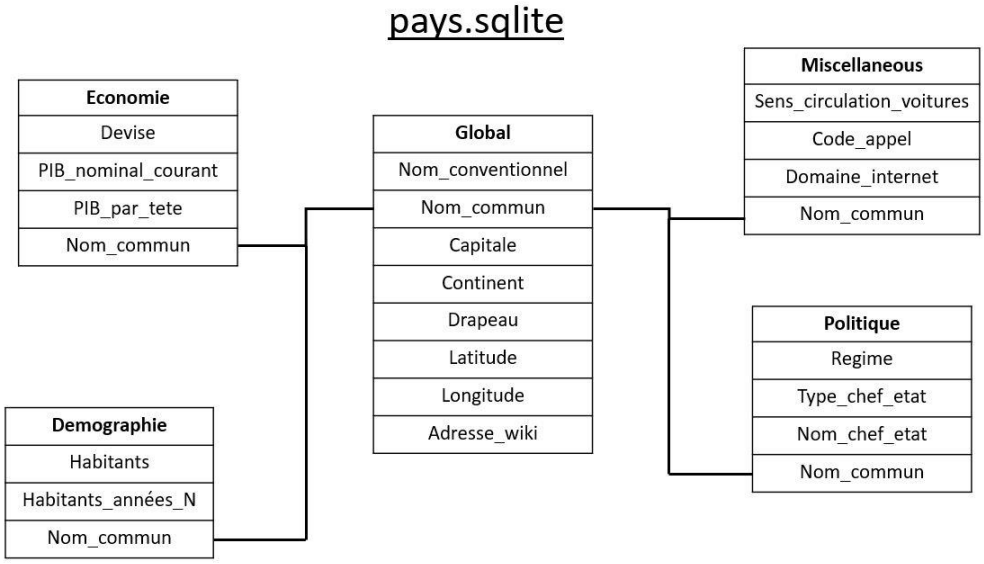


Figure 1 : Représentation de la base de données

II. Importation de la base de données dans le serveur

a. Présentation

Le fichier *import_database.py* permet de d'extraire les données de la base de données *pays.sqlite* et de créer une liste de dictionnaire utilisable par le serveur depuis le fichier *serveur_carte.py* .

Le principe de fonctionnement est assez simple. L'extraction et le renvoi se fait depuis la fonction *importation()* qui renvoie la liste des dictionnaires.

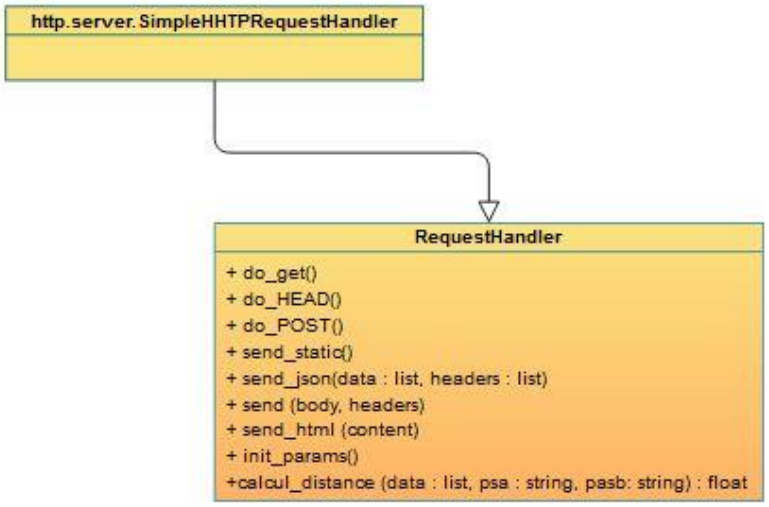
b. Fonctionnement

On initialise les clefs des différentes tables et on récupère les noms communs de tous les pays. En effet, la clef primaire de chaque table (et par lesquels non pourrions faire de *JOIN*) est le nom commun des pays.
Puis, pour chaque pays, on récupère ses données via les fonctions annexes *read_pays_table()* pour ensuite enregistrer ces données dans le dictionnaire de chaque pays.
Datalist, qui est renvoyée, est donc une liste de dictionnaire utilisée par le serveur.

III. Fonctionnement de l'application

a. Fonctionnement côté serveur

Notre serveur est un serveur python. On utilise la classe RequestHandler qui hérite de la classe http.server.SimpleHTTPRequestHandler, on crée de nouvelles méthodes et on en surcharge certaines.



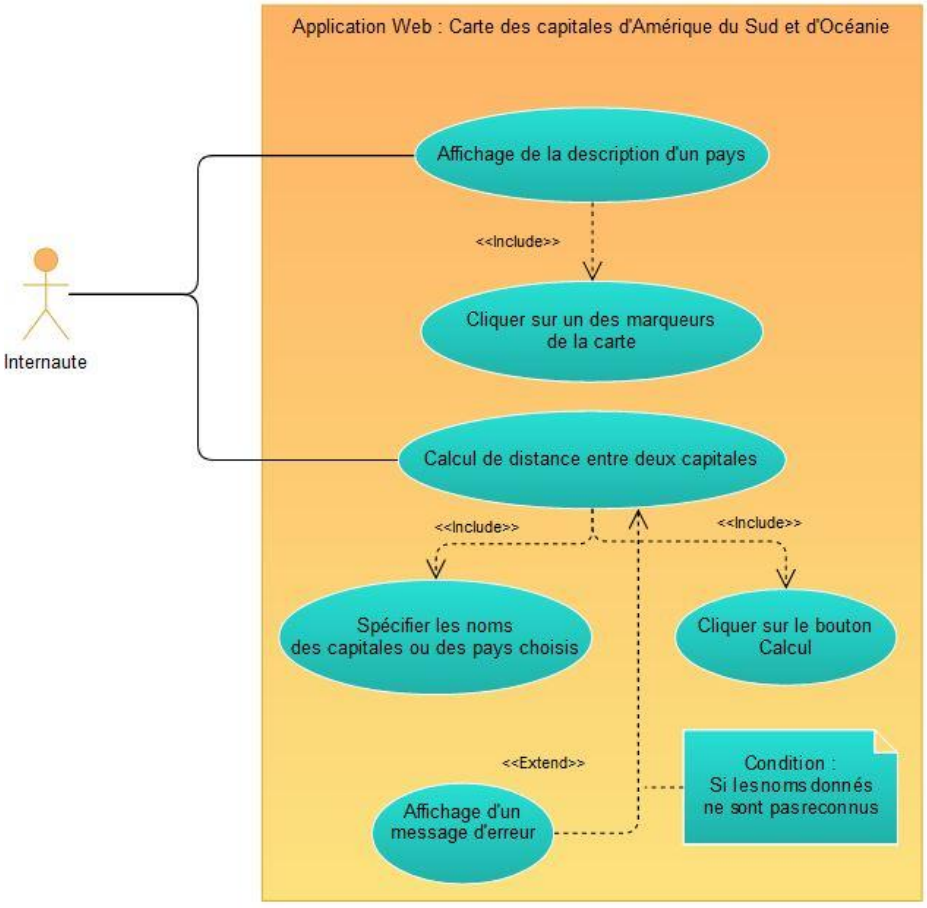
Le serveur fonctionne uniquement avec des méthodes GET.

-Pour afficher la carte et les repères sur celle-ci, le serveur reçoit la requête */location* via le chemin de l'url. Le serveur renvoie alors, l'ensemble des données au format *json*. Les données sont alors traitées en *javascript* pour afficher les repères sur la carte.

-Pour l'affichage des informations d'un pays en particulier, le serveur reçoit la requête */description/identifiant_pays*. L'identifiant pays est le code unique du pays, il permet de retrouver efficacement le pays dans l'ensemble des données. Les données du pays sont envoyées en format *json* pour être traitées en *javascript* pour ensuite afficher ces données à l'utilisateur.

-Lorsqu'une personne veut calculer la distance entre 2 capitales, la requête est */distance/pays_capitaleA/pays_capitaleB*. *Pays_capitaleA* et *Pays_capitaleB* sont envoyées dans la méthode calcul_distance(). Une réponse est déterminée selon la validité des arguments rentrés par l'utilisateur. Cette réponse est envoyée au client.

b. <u>Fonctionnement côté client</u>



Le client a plusieurs possibilités d'actions. Elle sont résumées dans le diagramme des cas d'utilisation ci-dessus.